AN INTEGRATIVE APPROACH TO ENGINEERING DATA AND AUTOMATIC PROJECT COORDINATION

Richard L. Segal
California Institute of Technology
Pasadena, California

5119:TM:83

Published at Conference on Automation Technology
Automation Technology Institute, Nov. 1983

AN INTEGRATIVE APPROACH TO ENGINEERING DATA AND AUTOMATIC PROJECT COORDINATION

Richard L. Segal
California Institute of Technology
Pasadena, California

(Published at Conference on Automation Technology
Automation Technology Institute, Nov. 1983)

Abstract

A new approach to integration of automation environments is necessary in order to overcome some fundamental limitations of current methods. The most common request for integration comes when an organization finds itself with many design systems and representations most of which cannot communicate directly. Solutions to this problem have required either a proliferation of translations between systems, or the provision of a central data base with which all applications must communicate. Another aspect of integration is the provision of mechanisms which keep track of the status of design. The working - release method along with version control techniques is well known but does not directly address the compatability issue and provides only a partial solution to status keeping. Another approach to integration is to exploit the underlying structure of an engineering project. This method can result in improvement in both realms of integration described above. The potential results of such an approach include the following, each of which is described in this paper. Firstly, this approach provides a new mechanism for providing design team independence and a better definition of design team responsibility. Secondly, the concept of inter-verification is introduced as a way of providing a continuum of representations which can be kept consistent through the project structure. Thirdly, automatic conflict detection and coordination mechanisms can be implemented, eliminating much of the delay usually associated with design changes. The final portion of this paper indicates some of the other areas influenced by the use of project structure in the computational environment. This includes, subdivision of verification processes, documentation systems, external object integration and automatic design systems.

1. ENVIRONMENT

An engineering organization contains an administrative structure for the purpose of managing and assigning resources, and a project structure associated with each ongoing product design. Engineering data is input, analyzed, output and communicated within and through the project structure. The project structure consists of "modules" each of which may contain "parts" which are instances of other modules. Since various modules may be in the process of design simultaneously, it is often necessary to make assumptions regarding the final form of parts of a module. These assumptions are called "requirements". The "design" of a module consists of manipulation and composition of parts. The task of a module design team is to provide design which will meet a particular set of "specifications." As specifications of a module become firmed up, the requirements placed on that module can change to reflect this process. Both requirements and specifications are constantly changing, first as the initial design team responsibilities are being worked out, and later as the details of the various designs take shape.

2. INTEGRITY

This project structure exists in every large engineering project in one form or another. By using this structure to organize the subdivision of data bases, we are suggesting a new approach to data base organization. Rather than a simple working - release set of data bases, each data base contains precisely the current representation of its design. Design team independence is accomplished through the RSD or Requirement, Specification, Design method. A module team performs its task by using requirements to represent actual parts. Specifications, though similar in form to requirements, are announced independently by each module team. Specifications are verified in terms of module design and requirements are verified in terms of part module specifications. Mechanisms to allow project refinement and keep track of the status of engineering data can be based on this structure. The working - release method of data base separation achieves independence, but does not maintain up to date interrelationships and does not guarantee the compatability of different working data sets.

We use the term verification to represent all forms of analysis, calculation, comparison and simulation procedures. Verification of specifications through whatever technique desired and verification of design correctness, which includes ordinary design system feedback and error detection, all takes place within a module. We call this "intra-verification. "Inter-verification" is the checking of correspondence between the requirements placed on a module and the actual specifications announced within that module. Any number of user modules may place requirements on a particular module. Each such requirement must be verified. By providing inter-verification for every kind of requirement, the current status of the complete engineering project is accessible at all times.

3. AUTOMATIC COORDINATION

Every kind of verification has a particular set of inputs which it can use. For example, specifications include statements like "the mass is between 50 and 60 kilograms" as well as "the circuit behavior is described by xxx" where xxx is a state machine description, or finally, "there is a socket at position 10.5,150,88", perhaps specified through a combination of graphics and text. Clearly, a specification may use any other element of module data in its verification. Verification of design, which includes operations like design rule checking and the detection of illegal manipulations at interactive design time, may only use information in the requirements, and information in other design elements. The latter is necessary when a composition of parts, each of which may themselves be rotated or otherwise manipulated or parameterized, is performed. Suffice it to say, that a complete and formal definition of the available input sets for each form of verification can be determined.

Each verification actually uses only some subset of its available input. This set is called the constituent set of a verification. If the constituent set of every verification is known, then it becomes possible to perform automatic design coordination. There are two aspects to this process. "Verification tracking" is the task of maintaining the current status of every data item in a project. This means that every time a data item is altered, each verification containing this item in its constituent set must be invalidated or reverification must occur. This set of verifications is called the

"track" of a data element. The second aspect of project coordination is called "notification". This is the task of informing designers when a change having a bearing on their design has been performed.

3.1 VERIFICATION TRACKING

A simple mechanism can be used to determine if an item is in the track of a particular design change. A procedure, called a "tracker", is provided for every kind of verification. A tracker is a suprisingly simple program, usually only a few lines of code, which just checks the type of the item which was altered and reports either positively or negatively that its associated verification has the altered item in its constituent set. For example, let us say there is a design item called "conn" which is used to join two objects together. There is a verification called "vcon" used to check the compatability of objects in a conn. A tracker for vcon, call it "tcon", would operate as follows. The parameters to a tracker include the item containing the verification to be considered and the the item which is being tracked. Tcon would first read the names of the objects being joined in conn. If the altered item is a connector type requirement on one of those two modules, then tcon returns a positive report, otherwise it returns negatively.

Trackers are all short similar programs usually far less complicated than the verification with which they are associated. However, if a verification is quite simple, rather than performing the usual check, the tracker can simply call the verification directly. This requires a third and fourth kind of response to be permitted from trackers. They are, that the tracker called the verification, which, in turn, either succeeded or failed.

The entire process of verification tracking consists of two steps. First, all the trackers which have the altered item in their available input set are called. Second, all the verifications for which a positive report is received (ie. that the item is in the track and was not reverified) are invalidated. Verification tracking should be an invisible operation which occurs whenever data is altered. It should be capable of dealing with a number of data items at one time, and should make effiecient use of networks, requesting all information needed from each module at one time.

3.2 NOTIFICATION

There are two aspects to notification. The first is negotiation notification. This occurs when a specification or a requirement is changed. When a module changes an element of its specifications, all users of that module are notified. When a module changes the requirements it places on a part module, the designers of that part module are notified.

The second kind of notification is validity notification. Any time a verification tracking operation occurs, each of the designers responsible for data in the track of the changed element must be notified of the action. They are told exactly which verifications have been invalidated or reverified.

4. CONFLICT RESOLUTION

The detection and resolution of design conflicts is one of the most important tasks of an engineering project. Typically, if a design

change in a low level module inherently conflicts with a high level management constraint, the problem might not be noticed for weeks or months. Usually there are regular project meetings at which the offending design data might be reported. Later, managers meet with summaries of each of their groups activities. There is no guarantee that the data in question will be reported at either of these meetings. Eventually the data will reach the party resonsible for performing the ill-fated analysis. By this time however, much work may have been wasted as the original error has been incorporated in later design.

In our environment, the management constraint would appear as a specification of a high level module. It would be verified based on requirements on other modules. These requirements must be matched by specifications on those parts which, in turn, must themselves be verified on the basis of their part requirements. The part module must achieve those requirements through its own specifications. Now, when a low level change occurs that violates the management constraint, it will appear as a tracking action on the specifications of that low level part module. The error is detected immediately.

The designer can either choose to change his/her design accordingly or to negotiate for some changes with users. This is done by changing the offending specification thus causing notification of the involved user module. The conflict could be resolved at this level, or perhaps more changes would be made causing the involvement of the highest level module. In any case, the current state of the design is always evident and delay and wasted effort are effectively eliminated.

Conflicts can appear as a result of the change of any element of the design. During interactive design procedures, engineers are used to being given feedback regarding the correctness of the operations they perform. This is a simple form of verification tracking and notification where the verifications are of design. By incorporating verification tracking in the underlying data base mechanisms, new dimensions are added to this process. By saying that verifications are associated with design items, we have in effect, explicitly separated verification from input process. In this example, feedback is still provided at design time, but now, those verifications are also associated with the resulting design data. Now, when changes are made to components of the design operation, those original verifications will be automatically invalidated. The designer is notified that reverification of the design operation needs to be performed. Instead of receiving design feedback only at design time, design validity is maintained through all aspects of design generation and change. Every kind of design operation can be involved in automatic conflict detection and resolution thus maintaining overall project integrity.

5. ONGOING RESEARCH

We have implemented the data base technique described above as well as the coordination mechanisms verification tracking and notification. The ACE (Automated Computer Engineering) system has been implemented within the ASK (A Simple Knowledgable) environment of Drs. Frederick and Bozena Thompson at the Califonia Institute of Technology. A small example of ACE activity is provided at the conclusion of this paper. Other areas affected by the project structure approach are also being investigated.

One interesting result of this approach is the suggestion that verification processes which usually operate on large projects should be subdivided along lines parallel to the project structure. Exploitation of hierarchy yields savings through redundancy elimination. Subdivision of a process into partial segments performed at times associated with the design process, results in the use of fractional data sets. That is, only a small portion of the data need be analyzed at any one time. Composition portions include the results of element verification. Less data at one time implies less disk usage and therefore better total execution time. Time savings may also accrue as a result of hiding partial verification behind the interactive design process.

Better automatic documentation systems are also sure to result from the project structure approach. A hierarchical decomposition of design is clearly available. Beyond this, we have also come up with a new use for the concept of transformation. Automatic transformations have typically only been written to translate between representations. Another kind of automatic transformation is one which extracts semantic information from the engineering data. A designer can choose a set of transformations to apply to his or her data in order to generate a particular form of feedback desired. For example, one might define a transformation called "boundary", another called "red" and another called "bearings" each of which extracts a particular set of information from the data base. By concatenating these transformations a designer can receive all kinds of feedback. For example, by joining all three in the order listed above one would get an answer to the question, "What are the boundaries of all red bearings?" This kind of transformation imposes a semantic interpretation upon an engineering data base. The interpretation of "red" or any other transformation is encapsulated in the program which is that transformation. By using the ASK natural language facilities [1] and implementing transformations as ASK operators, one can actually use English sentences to concatenate transformations as in the above example.

Verification can also make use of concatenated transformations. "Display the first mode of building X minus all rear elevators when struck by an earthquake of magnitude 5" makes use of the transformations "first mode", "minus", "elevators", "rear", "earthquake", and "magnitude 5". Another combination of the same transformations would yield entirely different results. This should indicate the range of possibilities of this approach to transformation. By incorporating this into the project structure environment even more powerful semantically guided operations should be possible.

Finally, the concept of parametric module can be extended to include automatic design production systems. We are in the process of formulating the basis of automatic design systems as they can interact with designers and the project structure. A whole new vision of automatic design is being revealed. In it, the instantiation process consists of derivation processes and decision processes. Derivation is the production of design from specification parameters and the respecification of modules based on changes to requirements. It involves producing data directly as well as producing changes to existing data based on user decisions. These changes in direction can be provided by the user after each level of derivation. Thus, automatic design becomes an interactive process in which design may occur as a top-down, bottom-up, or inside-out process.

6. CONCLUSIONS

The project structure existing within every engineering organization implicitly provides mechanisms of design team independence. Primarily, this is accomplished through the establishment of requirements on parts which represent the assumptions being made about the make-up of those modules. Exploitation of this structure in the organization of engineering data and systems yields a broad range of potential benefits. These potentials should be especially useful in engineering environments of considerable complexity where status keeping, integration, inter-communication and conflict resolution are serious and well established problems. We have at this time constructed preliminary prototypes for integrated circuit design, mechanical design, and software engineering environments. We hope to assist in the construction of an industrial environment in the near future.

7. SHORT EXAMPLE

```
>new copter fact   {starting a design dialogue}
>Part of the design of what module? block
>Element of specification? socket
>Side? right
>Position? 25,10,15
>Array dimensions? 8,2

The following new specification has been added:
1) Type is SOCKET.
   Direction is RIGHT.
   Positon is 25,10,15.
   Array size is 8,2.

 The verification of type "reqcon" of requirement 2
 of module "landing gear" has been reverified and SUCCEEDED.
 The verification of type "compcon" of specification 1
 of module "block" has been invalidated.
 The verification of type "reqcon" of requirement 1
 of module "rear body" has been reverified and FAILED.

----------

{excerpt from module design utility dialogue}
>Action? VS         {Verify a specification}
>Number of specification? 1
 Can't perform "compcom" verification until "nums" is valid.
 ***** Verification type "compcom" FAILED.
>Action? AVS        {Add a specification}
>Number of specification? 1
>Add what verification? nums
>Add what verificaiion?
 Changes saved.
>Action? VS
>Number of specification? 1
 ***** Verification type "nums" SUCCEEDED.
 ***** Verification type "compcom" SUCCEEDED.

----------

{Using ASK English}
>What is the type and designer of each part of XYZMOD?
```

part of XYZMOD	type	designer
inport	port	Segal
left wing	wing	Smith
outport	port	Segal
frame	casing	Segal

>What is the optimistic time, estimated time and pessimistic time of each task of frame?

task of frame	optimistic time	estimated time	pessimistic time
framplan	14	21	28
desorg	10	12	13
finjob	25	36	45
desdins	5	7	8

8. REFERENCES

[1] - Dr. F. B. Thompson and Dr. B. H. Thompson, "Introducing ASK, A Simple Knowledgeable System," Caltech Computer Science technical memorandum 5054, 1982.

[2] - Richard Segal, "Design Automation and Engineering Organization Structures," ACM SIGDA Newsletter, Sept. 1983.

[3] - William Beeby, "The Future of Integrated CAD/CAM Systems: The Boeing Perspective," IEEE Computer Graphics and Applications, Vol 2, #2, January 1982.

Biographical Sketch

Richard Segal was born in New York City in 1958. He moved to Albuquerque, New Mexico in 1971 and to California in 1975. There he received a B.S. in Information and Computer Science from the University of California at Irvine. He received a M.S. in computer science from the California Institute of Technology in Pasadena in 1981. Since then, under the direction of Dr. Frederick Thompson, Richard has been completing a Ph.D. program in Caltech's computer sciecnce department. The expected thesis title is "Computer System Integration in Engineering Design and Management." Richard's background includes fourteen years programming experience, integrated circuit design, design tools, and artificial intelligence. He is a member of the ACM, IEEE, SCS, and AAAI.